INFORMATION TECHNOLOGY LABORATORY

Understanding Consistency Maintenance in Service Discovery Architectures in Response to Message Loss

Chris Dabrowski, Kevin Mills, Jesse Elder

AMC 2002

Edinburgh, Scotland

July 23, 2002

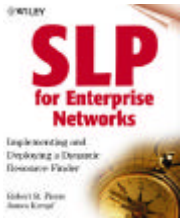
NIST
National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

Dynamic discovery protocols in essence…

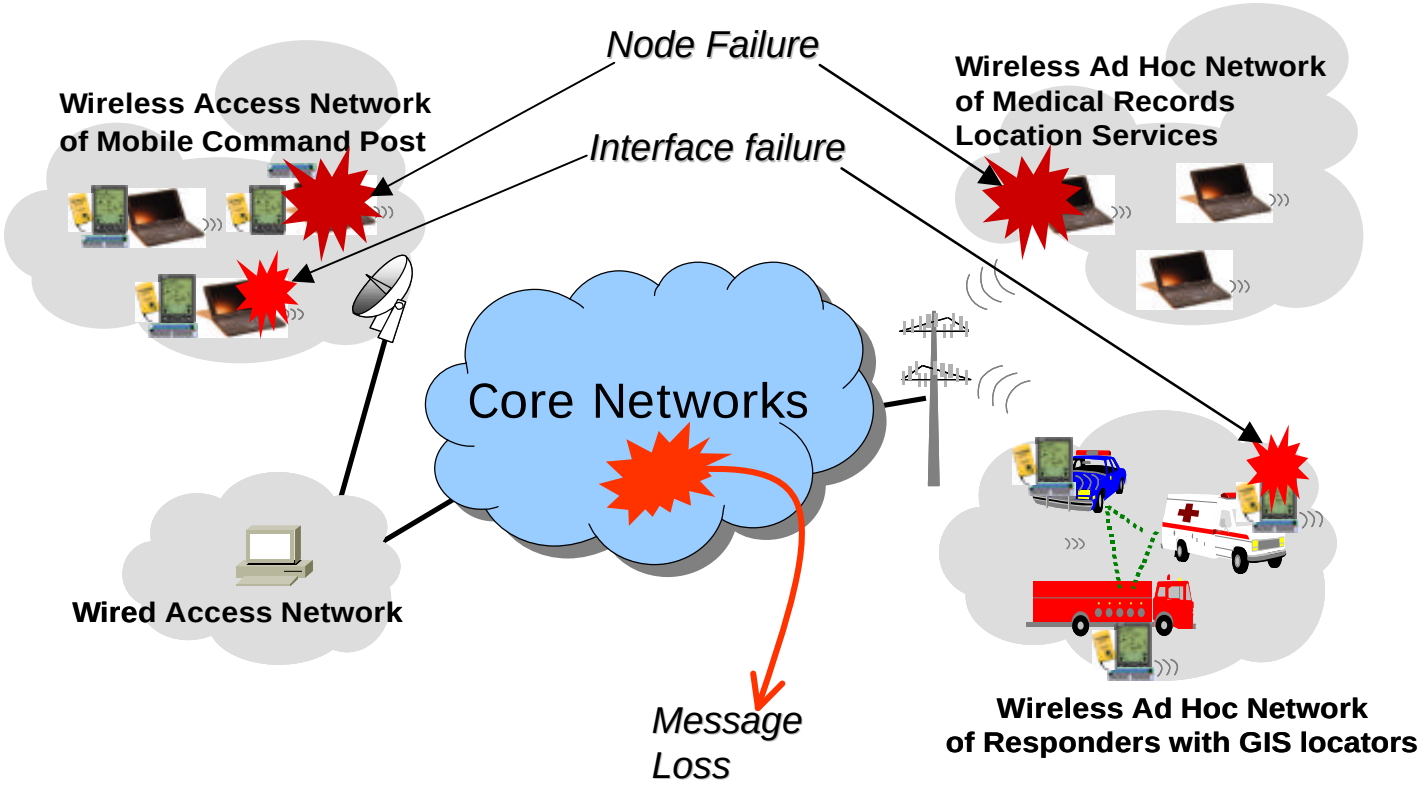
enable ***distributed software components***

(1) to ***discover*** each other without prior arrangement,
(2) to ***express*** opportunities for collaboration,
(3) to ***compose*** themselves into larger collections that cooperate to meet an application need, and
(4) to ***detect and adapt*** to failures.

Some examples:

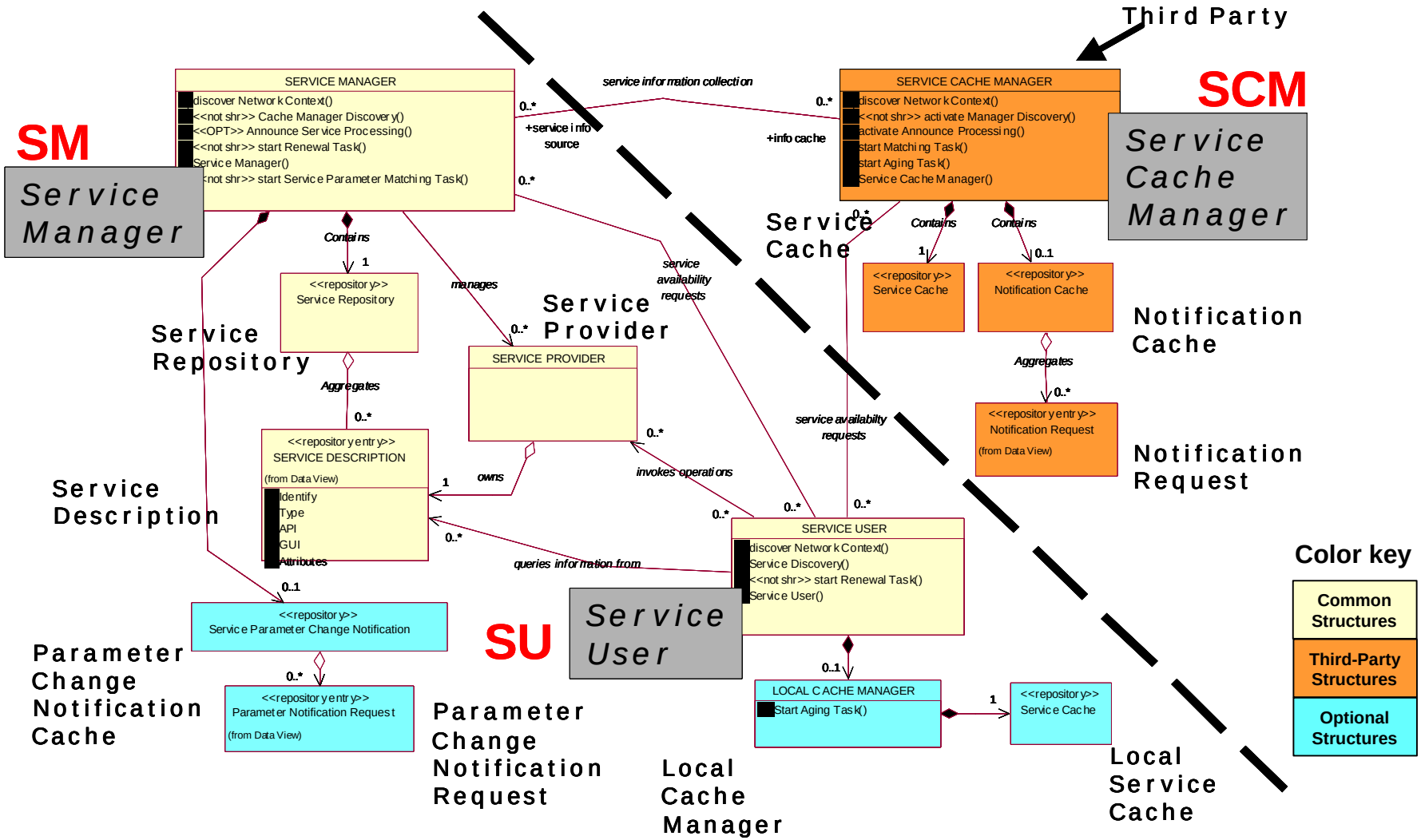
JINI **3-Party Design**	UPnP FORUM **2-Party Design**	 **Adaptive 2/3-Party Design**
The Salutation Consortium **Vertically Integrated 3-Party Design**	HAVi **Network-Dependent 3-Party Design**	Bluetooth™ **Network-Dependent 2-Party Design**

Motivation: Failure in Hostile and Volatile Conditions

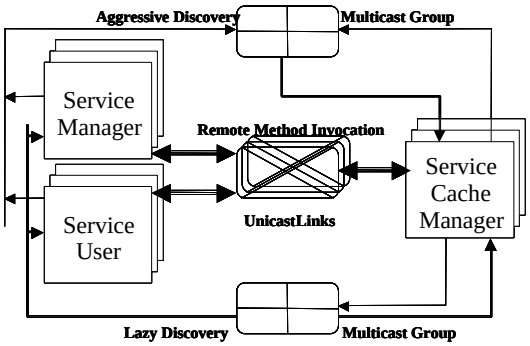


- **Focus of study: comparative robustness of different combinations of protocol architectures, topologies, and consistency maintenance mechanisms during message loss.**

General Architecture for Service Discovery Systems



Modeling and Analysis Approach

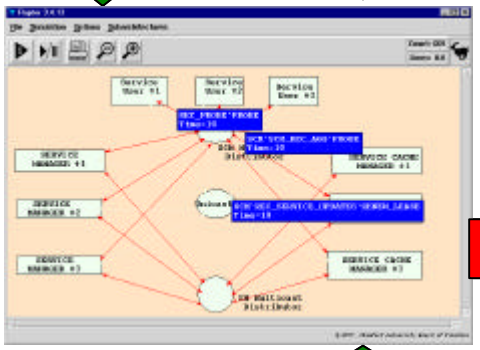


Topology

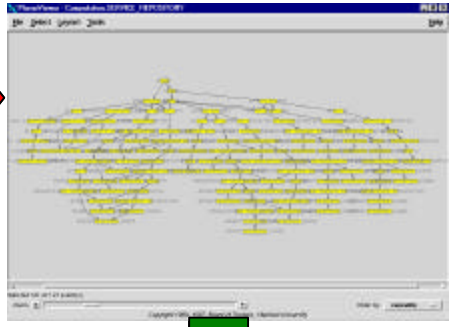
Scenario

Time	Command	Parameters
5	NodeFail	SM4
5	LinkFail	SCM1 SM4
10	GroupJoin	SM4 GROUP1
10	FindService	SU8 5 1 2 S XYZ ALL
50	AddService	SM4 SCM3 T ATT API GUI 20 30

Behavior Model



Execute with Rapide



```
-- ***************************************************
-- ** 3.3  DIRECTED DISCOVERY CLIENT INTERFACE      **
-- ***************************************************
-- This is used by all JINI entities in directed
-- discovery mode.  It is part of the SCM_Discovery
-- Module. Sends Unicast messages to  SCMs on list of
-- SCMS to be discovered until all SCMS are found.
-- Receives updates from SCM DB of discovered SCMs and
-- removes SCMs accordingly
-- NOTE: Failure and recovery behavior are not
-- yet defined and need reviw.
TYPE Directed_Discovery_Client
  (SourceID : IP_Address; InSCMsToDiscover : SCMList; StartOption : DD_Code;
   InRequestInterval : TimeUnit; InMaxNumTries : integer; InPV : ProtocolVersion)
IS INTERFACE
SERVICE DDC_SEND_DIR    : DIRECTED_2_STEP_PROTOCOL;
SERVICE DISC_MODES      : dual SCM_DISCOVERY_MODES;
SERVICE DD_SCM_Update   : DD_SCM_Update;
SERVICE SCM_Update      : SCM_Update;
SERVICE DB_Update       : dual DB_Update;
SERVICE NODE_FAILURES: NODE_FAILURES; -- events for failure and recovery.
ACTION
 IN Send_Requests(),
    BeginDirectedDiscovery();
BEHAVIOR
   action animation_Iam (name: string);
   MySourceID        : VAR IP_Address;
   PV                : VAR ProtocolVersion;
```

Consistency Conditions

For All (SM, SD, SCM):
(SM, SD) IsEleme ntOf SCM registered -services (CC1)
implies SCM IsElementOf SM discovered -SCMs

For All (SM, SD, SCM):
SCM IsEl ementOf SM discovered -SCMs & (CC2)
(SD) IsElementOf SM managed -services
implies (SM, SD) IsElementO f SCM registered -services

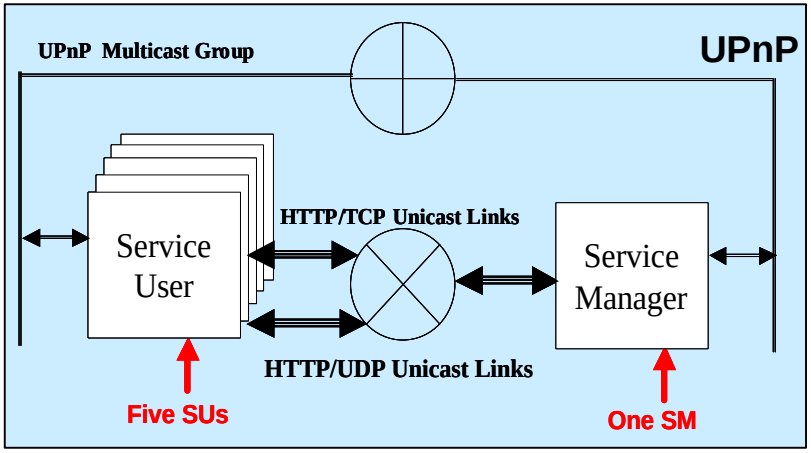
For All (SM, SD, SCM):
SCM IsElementOf SM discovered -SCMs & (CC3)
(SM, SD) IsElementOf SCM registered -services &
NOT (SCM IsElementOf SM persistent -list)
implies Intersection (SM GroupsToJoin, SCM GroupsMemberOf)

For All (SM, SD, SCM, SU, NR):
(SU, NR) IsElementOf SCM requested -notifications & (CC4)
(SM, SD) IsElement Of SCM registered -services &
Matches((SM, SD), (SU,NR))
implies (SM, SD) IsElementOf SU matched -services

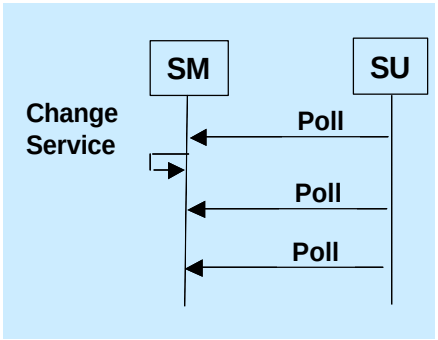
Analyze POSETs

Use metrics to Assess Correctness & Performance

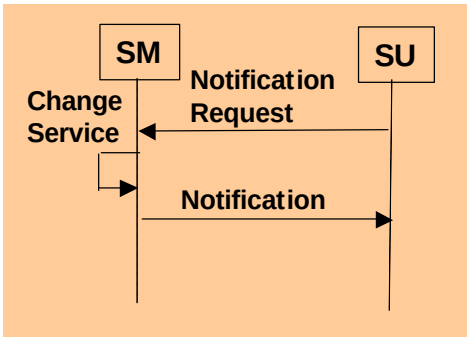
How do various service discovery architectures, topologies, and consistency-maintenance mechanisms perform under deadline during message loss conditions?

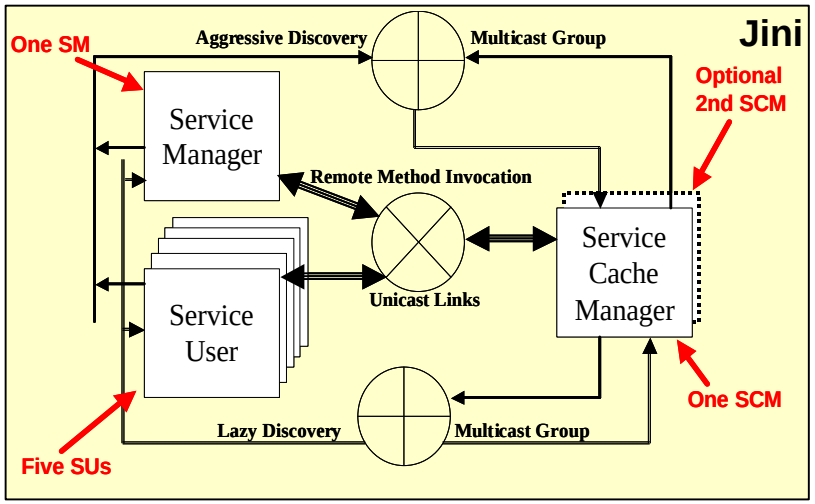


Two-Party Polling

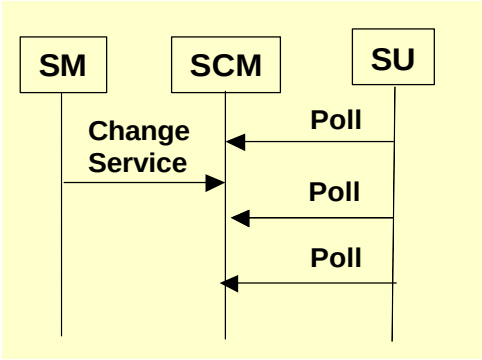


Two-Party Notification

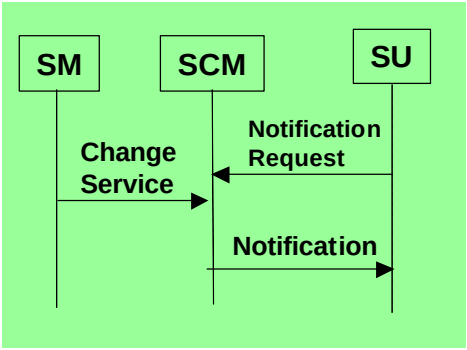




Three-Party Polling

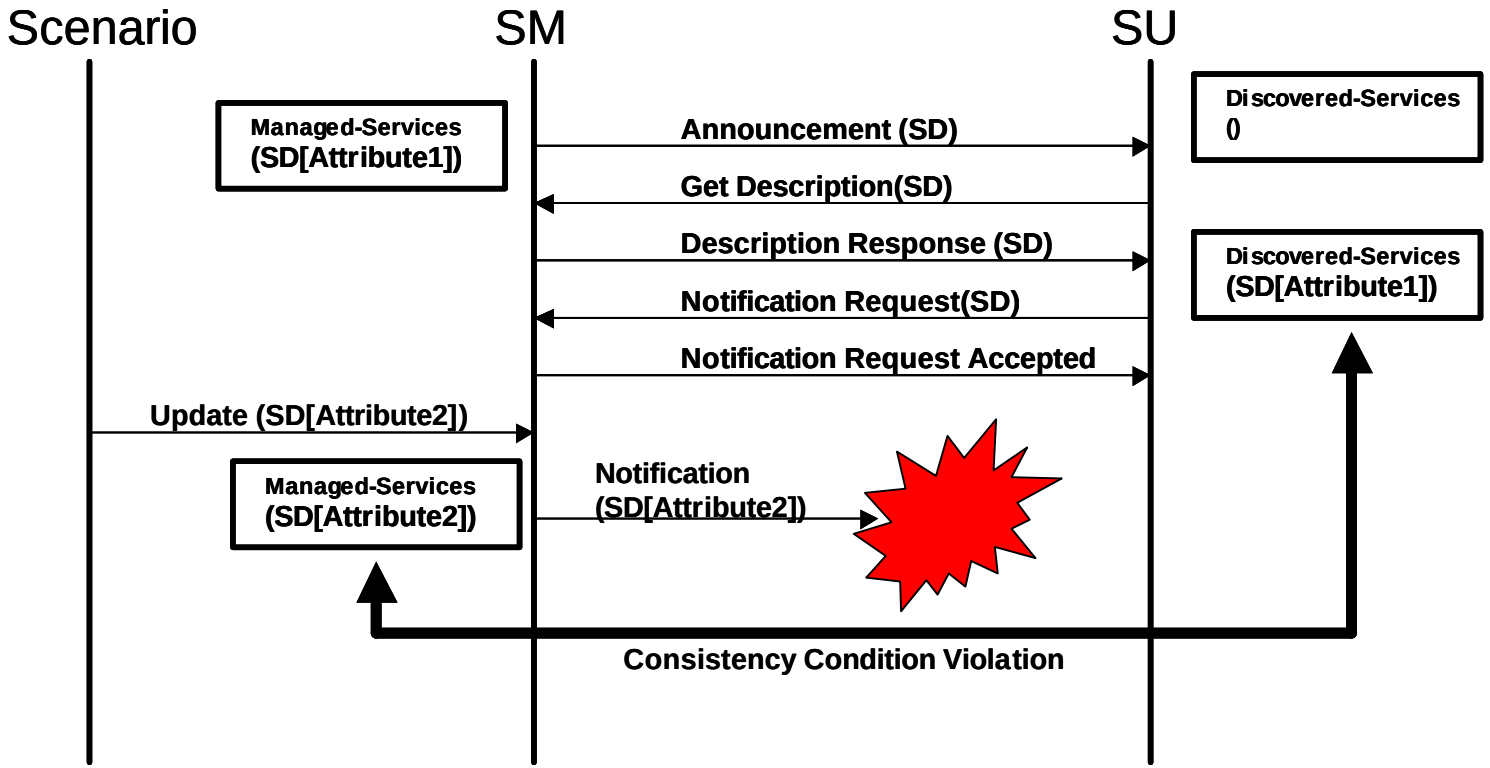


Three-Party Notification



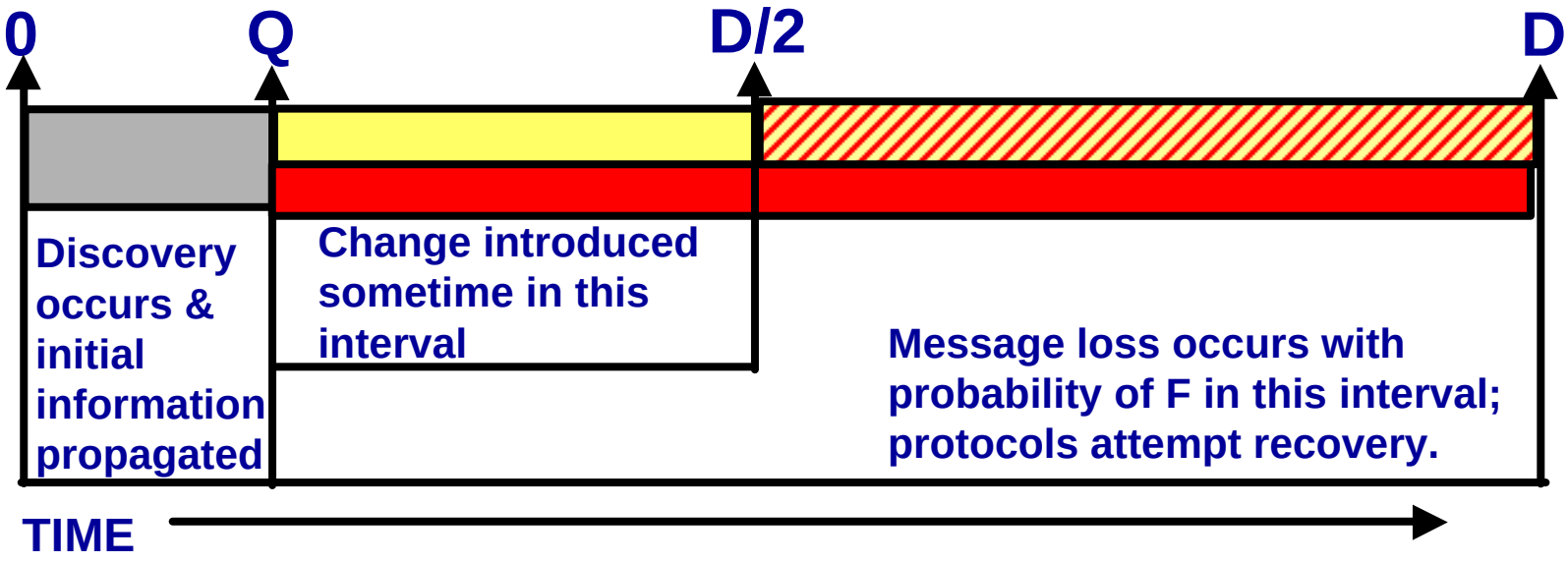
Monitoring Consistency

For All **(SM, SU, SD):**
(SM, SD [Attributes1]) *IsElementOf* SU discovered-services
SD [Attributes2] *IsElementOf* SM managed-services
implies **Attributes1 = Attributes2**



How well does the system restore consistency after message loss?

Modeling Message Loss



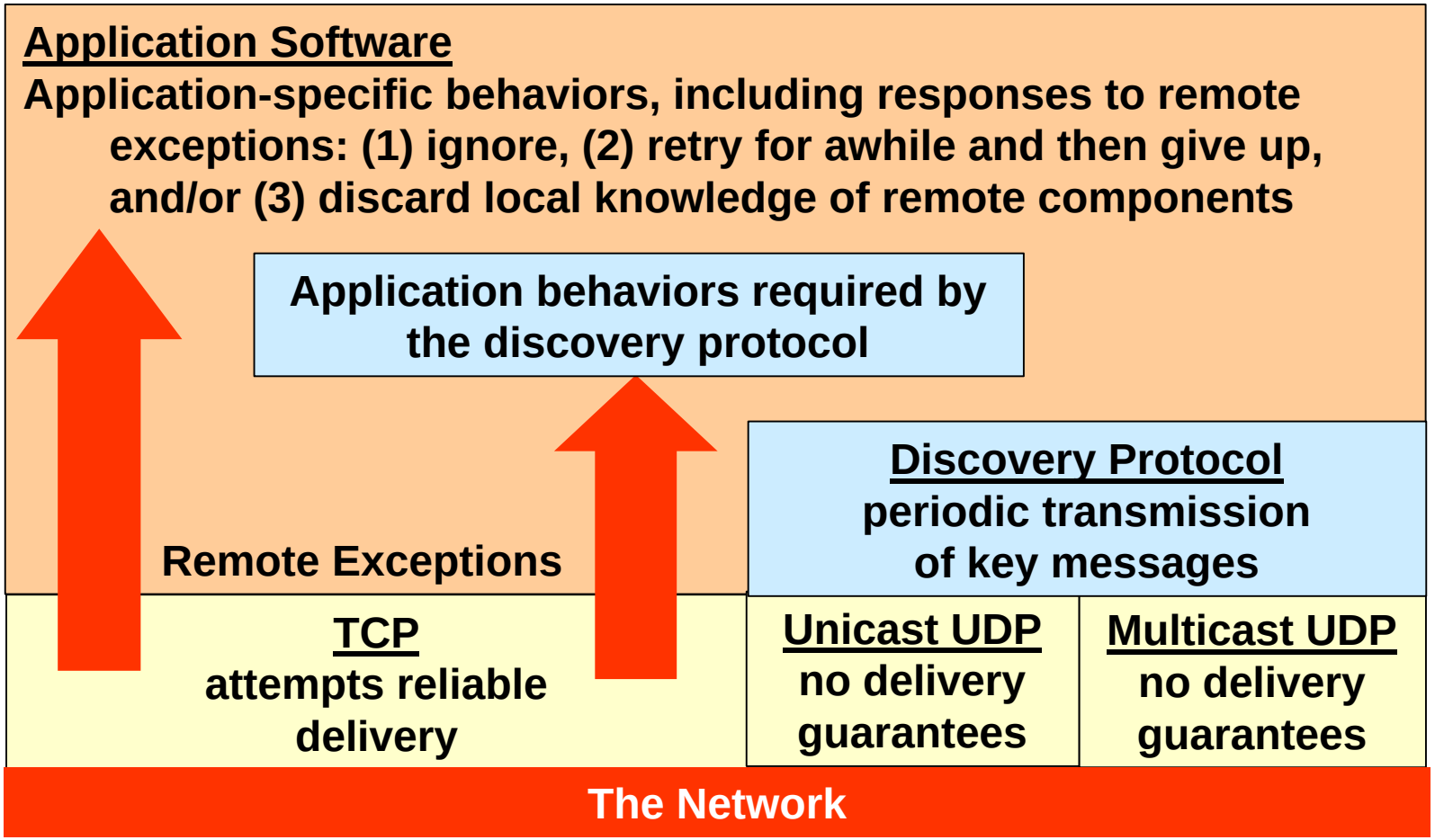
Random Processes

1. **Choose a time to introduce the change [uniform(Q, D/2)]**
2. **For each message transmission, determine if message is lost using F**

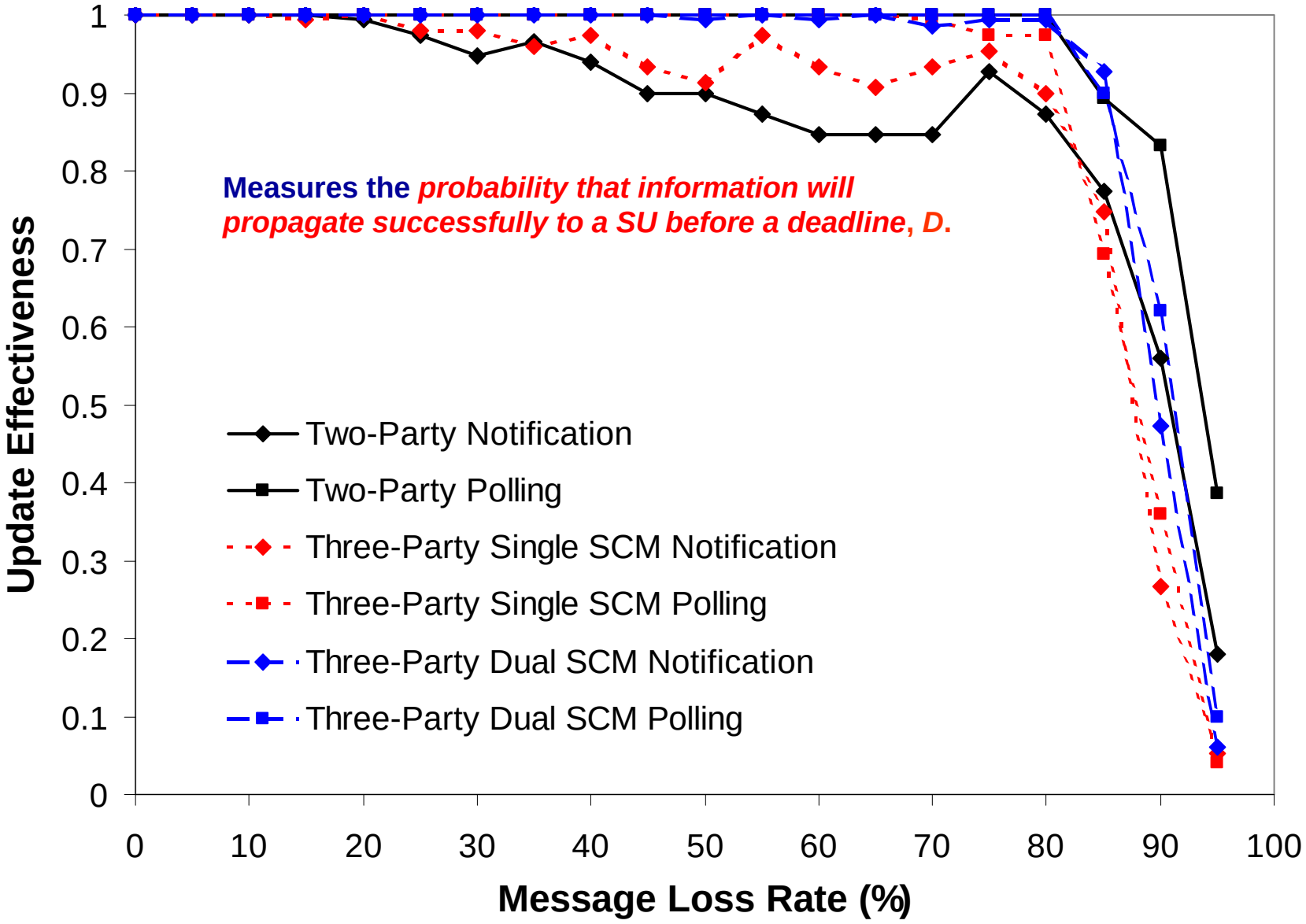
Q = end of quiescent period (100 s in our experiment)
D = propagation deadline (5400 s in our experiment)
F = message loss rate (variable from 0% - 95% in 5% increments in our experiment)

Division of Failure Recovery Responsibilities:

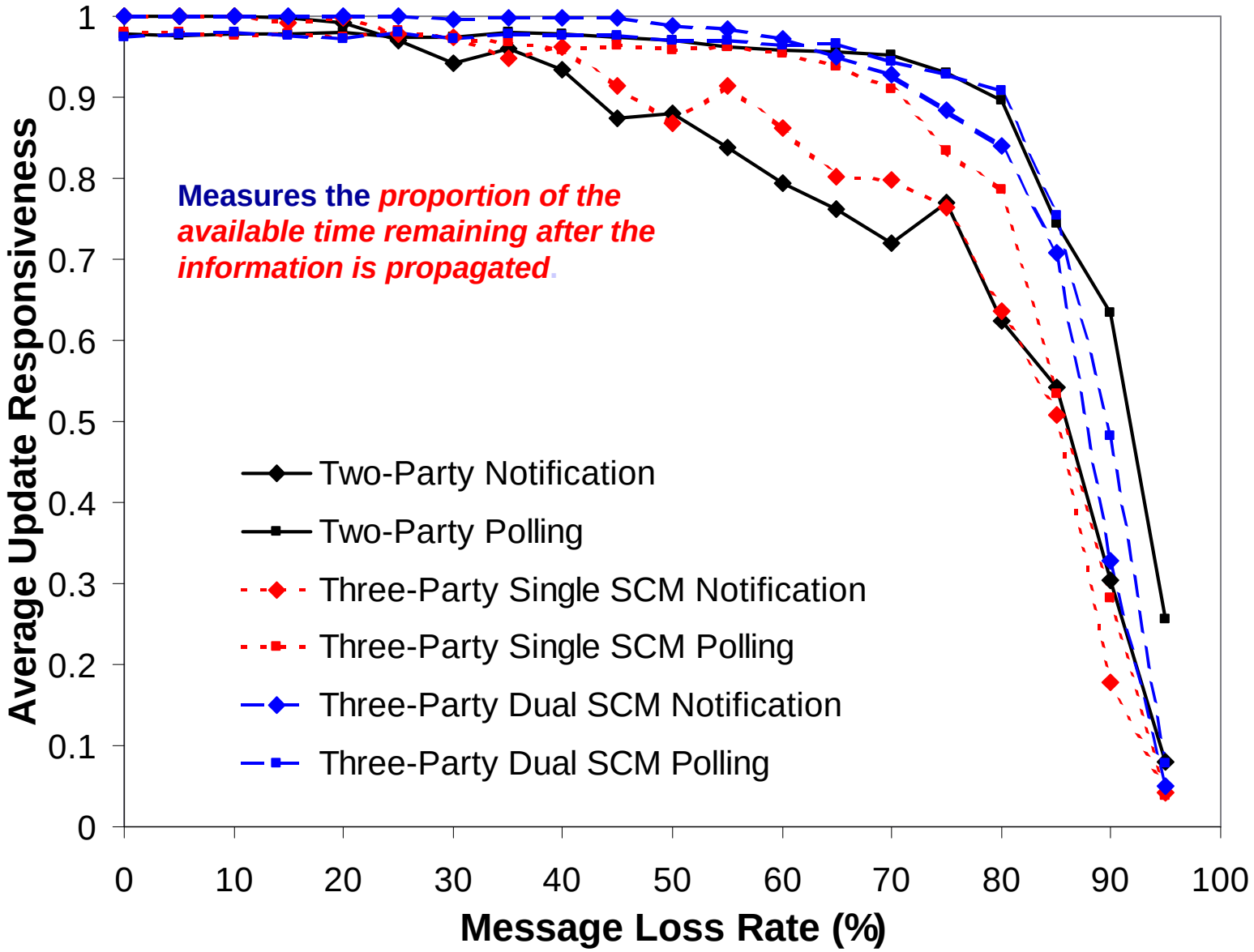
Communication Protocol - *Discovery Protocol* - *Application Software*



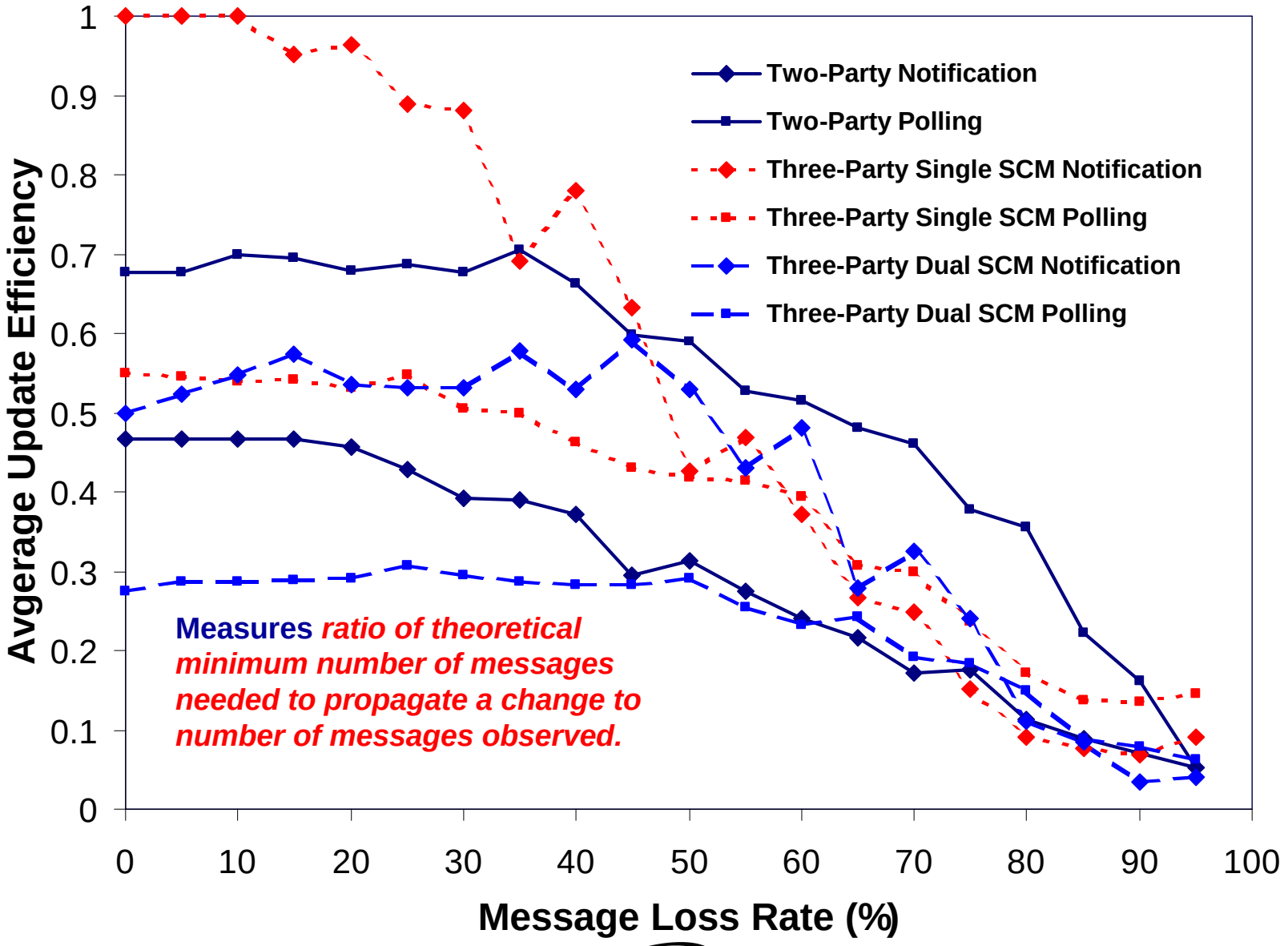
Update Effectiveness UPnP (2-Party) vs. Jini (3-Party)



Update Responsiveness UPnP (2-Party) vs. Jini (3-Party)



Update Efficiency UPnP (2-Party) vs. Jini (3-Party)



Conclusions

- **Executable architectural models represent essential complexity and reveal collective dynamics – leading to valuable insights into performance of middleware services during message loss**
  - paper specifications do not represent dynamics very well
  - reference implementations exhibit substantial incidental complexity
- **A single architectural model can be analyzed for behavioral, performance, and logical properties**
  - limits errors and inconsistencies that can creep in when using multiple models to represent different facets of a design
- **Due to effect of recovery strategies, 2- and 3-party discovery architectures exhibit similar robustness during message loss, but**
  - Overall, polling provides slightly better effectiveness and responsiveness but lowers efficiency, but with significant exceptions.
  - Sole reliance on TCP retransmissions to recover notifications leads to an unexpected reduction in update effectiveness at lower message loss rates, which rises at higher rates as recovery strategies come into play (most pronounced for UPnP; Jini has additional SM behaviors that compensate).